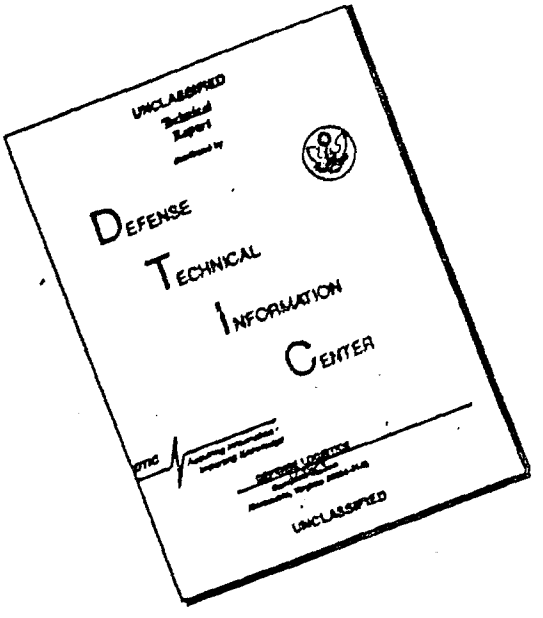
Domain Decomposition with Local Mesh Refinement

William D. Gropp† and David E. Keyes‡

Research Report YALEU/DCS/RR-726
August 1989

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

We describe a preconditioned Krylov iterative algorithm based on domain decomposition for implicit linear systems arising from partial differential equation problems which require local mesh refinement. In order to keep data structures as simple as possible for parallel computing applications, the fundamental computational unit in the algorithm is a subregion of the domain spanned by a locally uniform tensor-product grid, suggestively called a tile. This is in contrast to local refinement techniques whose fundamental computational unit is a grid at a given level of refinement. The bookkeeping requirements of such algorithms are potentially substantial, since consistency of data must be enforced at points of space which may belong several different grids, and furthermore, the grids are not necessarily of tensor-product type, but more generally, unions thereof. The tile-based domain decomposition approach condenses the number of levels in consideration at each point of the domain to two: a global coarse grid defined by tile vertices only and a local fine grid, where the degree of resolution of the fine grid can vary from tile to tile. Experimentally, it is shown herein that one global level and one local level provide sufficient flexibility to handle a diverse collection of two-dimensional problems which include irregular regions, non simply-connected regions, non-self-adjoint operators, mixed boundary conditions, non-smooth coefficients, or non-smooth solutions. We employ from 1 to 1024 tiles on problems containing up to 16K degrees of freedom. Though motivated by local refinement and parallel processing applications, benchmark serial implementations of the tile-based algorithm on uniform grids produce iteration counts and execution times which are competitive with those of traditional global preconditionings.

Domain Decomposition with Local Mesh Refinement

William D. Gropp† and David E. Keyes‡

Research Report YALEU/DCS/RR-726
August 1989

† Department of Computer Science, Yale University, New Haven, CT 06520. The work of this author was supported in part by the Office of Naval Research under contract N00014-86-K-0310 and the National Science Foundation under contract number DCR 8521451.

‡ Department of Mechanical Engineering, Yale University, New Haven, CT 06520. The work of this author was supported in part by the National Science Foundation under contract number ECT-8707109.

1. Introduction

The combination of domain decomposition with preconditioned iterative methods provides a framework which extends the usefulness of numerical techniques for certain special partial differential equation problems to those of more general structure. Non-smooth features, non-separable geometries, or massive sizes of practical problems limit the application of many "standard" numerical techniques. Direct methods are rapidly defeated by problem size. "Fast" methods which take advantage of special coefficient and grid structure often do not apply globally. Iterative methods often depend for efficient implementation on regular grids which, if global in extent, are inconsistent with accurate and economical resolution of the physics of the problem. However, the domains of problems with these features can often be decomposed into smaller subdomains of simpler structure, increasing the utility of extant software libraries, particularly as components of preconditioners. Moreover, the domain decomposition can be made to produce a transparent mapping of many problems onto medium-scale parallel computers. Our primary focus in this paper is the incorporation of spatially-varying mesh refinement requirements into a finite-difference-based domain decomposition algorithm. We illustrate the convergence behavior of the algorithm on a variety of two-dimensional elliptic PDE problems, including non-self-adjoint, non-separable geometry cases. We also point out features of the method which are relevant to a parallel implementation but defer the corresponding complexity analysis to a subsequent companion paper.

Many PDE problems which are "large" in the discrete sense are so because the continuous problems from which they are generated require resolution of several different length scales for the production of a meaningful solution. The value of compromising between the extremes of globally uniform refinement, which leads to simple and usually vectorizable algorithms but wastes time and memory, and pointwise adaptive refinement, which minimizes the discrete problem size for a given accuracy requirement but leads to complicated data structures, has been recognized for some time and described in contexts too numerous to acknowledge fairly. Locally Uniform Mesh Refinement (LUMR) characterizes one such class of discretizations, based on composites of highly structured subgrids. Many treatments of LUMR in the literature pertain to explicit methods for transient problems, a class with its own advantages (see [3] and references therein) and limitations [39] which is somewhat distinct from ours. Implicit treatments of locally regular refinement for elliptic problems include approaches arising out of classical multigrid (see [31] and references therein), a nonconforming spectral technique [30], and methods rooted in iterative substructuring for finite element problems [5].

Computationally practical locally uniform grids are usually expressible as the union of a coarse uniform tensor-product grid covering the entire domain with one or more refined tensor-product grids defined over subregions, including the possibility of multiple, nested levels. Generalizations of this within the LUMR framework include allowing the grids at any particular level of refinement to themselves be the union of tensor-product subgrids, and reinterpreting "uniform" as "quasi-uniform" to allow general curvilinear coordinates for custom body- or solution-fitting. We select for consideration a rather restricted form of LUMR in which refinement occurs exclusively within complete cells of a quasi-uniform coarse grid, as described in section 2 below.

The goal of the present contribution is an LUMR methodology with starkly simple data structures, for efficient portability to a variety of parallel machines. It borrows from the mesh refinement and domain decomposition literature and from the authors' own experience in these areas and in parallel computation [20, 22, 28]. In our pursuit of convenience and overall parallel performance, in which we include both absolute speedup and efficiency, we are ready, potentially, to compromise "optimality" as defined by conventional serial computing measures. For example, by refining only in units of full coarse grid cells, we may impose a tendency towards refinement in regions where it

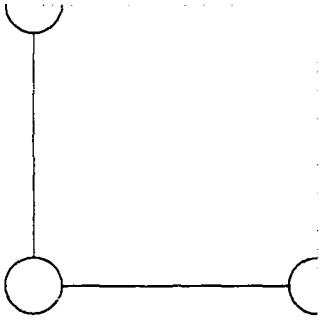


Figure 1: The anatomy of a tile. Unless closed by a physical boundary, a tile is open along its high-x and high-y perimeter.

would be unnecessary from a truncation error point of view alone. As another example, our convergence rate is dependent upon a coarse grid resolution which may be chosen with criteria beyond convergence rate in view, such as the balance of work among multiple processors. Fortunately, the methodology survives such compromises and is even sequentially advantageous in many problems.

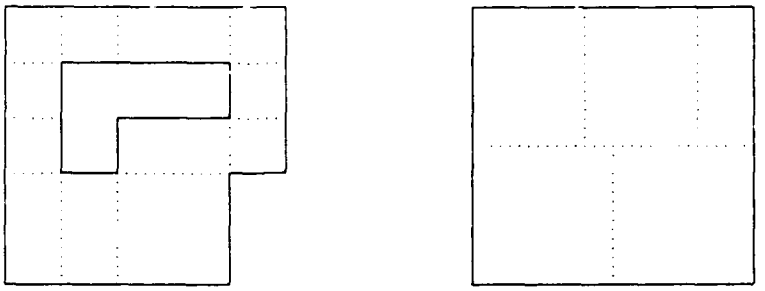
The domain decomposition algorithms we employ (section 3) involve "nearly" parallel preconditioners in conjunction with generalized minimum residual (GMRES) iteration, a non-stationary method not dependent upon operator symmetry. In two dimensions, the preconditioner involves three separate phases: a global coarse grid solve, independent solves along interfaces between subdomains, and independent solves in the subdomain interiors. The global coarse grid solve, which we do directly, is an essential feature as it provides the only global exchange of information in the preconditioner itself. We will compare alternative formulations of the more negotiable interface and subdomain solves.

The main body of the paper is the collection of numerical experiments on two-dimensional elliptic boundary value problems in section 4. The experiments include standard model problems, "L"-shaped, "T"-shaped, and non-simply-connected regions, non-self-adjoint operators, mixed boundary conditions, and problems with non-smooth coefficients or non-smooth solutions. We use from 1 to 1024 coarse grid elements on problems containing up to 16K degrees of freedom. Among our findings is that the interface probe preconditioning advocated in our earlier work on convective-diffusive systems with stripwise decompositions [28] does not perform as well on decompositions with internal vertices as the much simpler tangential operator preconditioning. We also demonstrate that incomplete factorizations are not effective subdomain interior preconditioners, relative to exact subdomain solves, once the subdomains become sufficiently narrow.

2. Mesh refinement by tiles

In this section we describe a simple mesh refinement philosophy based on a regular tessellation of the global domain into subdomains which we call "tiles" in two dimensions. Mathematically, a tile is the tensor-product of half-open intervals in each coordinate direction, except that a tile abutting a physical boundary along what would ordinarily be one of its open edges is closed along that edge. Each tile possesses its own tensor-product discretized interior, at least two of its four sides, and at least one of its four corners. Although the specific convention is arbitrary, we assume for definiteness that in its own local right-handed coordinate system, each tile contains its origin and its x and y axes (see Figure 1).

In contrast to physical boundary segments, we refer to the artificial decomposition-induced boundaries of the tiles as "interfaces". We refer to the points at the intersection of all boundaries,



(a) (b)

Figure 2: Sample Tessellations. (a) is permissible, (b) is not.

physical or artificial, as "cross-points". We require that the cross-points be embeddable in a tensor-product global quasi-uniform coarse grid, from which only points lying exterior to the (possibly multiply connected) boundary are missing. This rules out irregular tiling patterns such as in Figure 2b. However, there is no requirement that the domain itself be of tensor-product type: the decomposition in Figure 2a is permissible.

Associated with each tile is the data defined over a quasi-uniform grid covering its portion of the domain and a set of operators for executing its block-row portions of the preconditioner solve to be described later. In our object-oriented approach, these operators can potentially be of different types for different tiles. For computational convenience, we assume throughout that the grids covering individual tiles are derived from the coarse grid of cross-points through refinement in ratios of powers of two. We can therefore indicate refinement levels using the graphical shorthand of Figure 10 where the integer indicates the logarithm of the refinement ratio.

2.1. Tile-tile interfaces

In order to minimize restrictions on the structure of adjacent tiles (and to eliminate redundant communication between tiles in a multiprocessor implementation, in which different tiles might be assigned to different processors), each tile stores and maintains, in addition to its own data, the data associated with a buffer region of phantom points equal in width to one-half of that of its associated finite difference stencil (see Figure 3). Excluding the redundant phantom points, each point of the domain is uniquely associated with a single tile.

Data at the phantom points is supplied in a manner dependent upon the internal structure and refinement ratios of the adjacent tiles in question. A finer tile obtains bi-quadratically interpolated data from its coarser neighbor. Since the problems studied herein involve second-order operators, this allows the use of conventional finite difference techniques in generating the difference equations at the subdomain interfaces. Bi-linear interpolation alone would limit the potential accuracy of a second-order differencing scheme, as observed in some preliminary experiments. We note that such a difference scheme does not guarantee discrete flux conservation. Our focus herein is simply on the solution of a consistent set of discrete equations. More careful attention to the discretization has already been given in the context of locally regular refinement in [19].

All of our examples employ strictly uniform local grids. Although this is not a necessary restriction of the method, this simplifies the exchange of data between adjacent tiles.

The coarse grid system obtains its data by simple (unweighted) injection. That is, the value at the point in the finer neighboring tile that lies on the coarse grid stencil is used for the coarse grid point. A weighted averaging could be employed to preserve operator symmetry, if that were

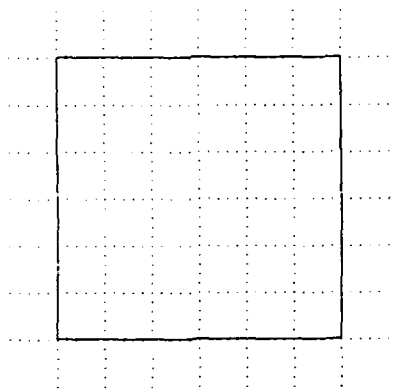


Figure 3: Sample tile, showing the computational buffer region required for the standard five-point stencil.

necessary for other reasons, for instance, conjugate gradient iteration of a self-adjoint problem. A finite element discretization with transition elements along the interface would unambiguously deliver the appropriate weighting coefficients in this case.

The selection of refinement criteria is a much studied, yet still open problem; see [2] and [26] for a sampling of work in this area. The refinement criteria, however, are orthogonal to the equation-solving aspect considered herein, except to the extent that a part of the computational work required by one of these tasks may be a by-product of the other. Some issues in refinement criteria will be discussed in a subsequent report [23]. For present purposes, we give one example with a smooth solution but non-smooth coefficients and others with smooth coefficients but a non-smooth solution. In these examples, "good" refinement strategies can be done "by hand".

In general, tile interfaces can be the site of changes in the discretization besides just the refinement level. For instance, the discrete stencil can change order at interfaces. Even the form of the operators or their number can change at interfaces while still preserving the subdomain uniformity required for efficient subdomain solution algorithms. As a motivational example, a reacting flow problem frequently consists of large regions in which there is only transport of mass, momentum, and thermal energy but no reaction among constituents of known composition, to all adequate orders of approximation. In other regions it is essential to retain composition variables, because they diffuse differentially, and in a subset of these, reaction terms must also be retained in the equations. To accommodate such generality, the routines that pack the buffer regions are responsible for providing the necessary mappings.

2.2. Physical boundaries

For generality, the equations for the physical boundaries are incorporated into the overall system matrix, including Dirichlet conditions. Our implementation allows inhomogeneous Robin boundary conditions at all boundary points, namely,

$$a(x,y)\frac{\partial u}{\partial n} + b(x,y)u = c(x,y).$$

Both first- and second-order one-sided difference approximations to the normal derivative term are employed. The second-order approximation is used in the actual operator, and the first-order is used in the preconditioners (to preserve uniformity of the bandwidth of the matrices used in the preconditioning). Though tempting in their simplicity, Dirichlet boundary conditions alone in the preconditioner were found to perform poorly in practice, in accord with expectation from the theory in [33] and references therein.

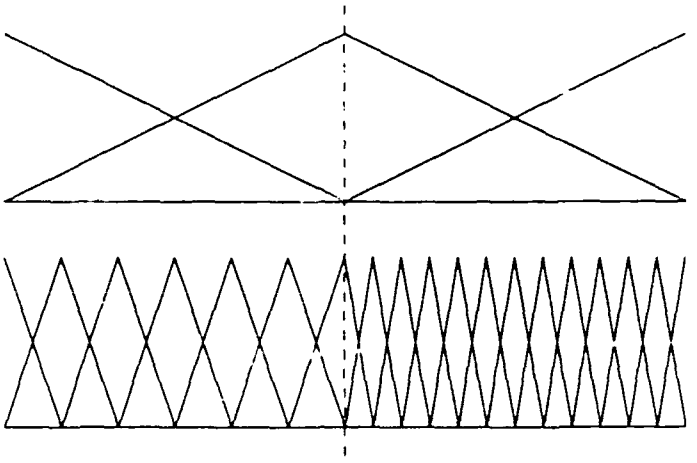


Figure 4: One-dimensional schematic of the tile basis functions.

2.3. Comparison with other approaches

In contrast to multi-level approaches in which the fundamental computational unit is a grid at a given level, our fundamental computational unit is a subregion of the domain. The present approach requires only one grid which possesses connectivity with arbitrarily distant regions of the domain, namely the coarsest one. In the framework of the hierarchical basis function technique [43], we have simply a two-level hierarchy, but the the higher level may be different in different subregions. Figure 4 gives a one-dimensional illustration. This admittedly represents a severe condensation of the range of intermediate scales present in multi-level local uniform refinement, on which much of the asymptotic convergence theory is based. Tiles are much closer to being the software equivalent of the "geometry-defining processors" (GDPs) of Dewey and Patera [14].

The tile approach is also similar to the additive Schwartz method [16, 41] and the techniques of [6] in its reliance upon just a single domain-spanning grid. The main difference between these techniques and the tile approach is in the treatment of the interfacial degrees of freedom. In the additive Schwarz technique, interior problems are solved on extended overlapped subdomains, of which the interfacial degrees of freedom are interior points and thus demand no special consideration. In [6], good preconditioners for the interfacial degrees of freedom are derived theoretically, for self-adjoint operators. Optimal algebraic convergence (independent of degree of refinement) has been proved for both classes of algorithms in [18] and [5], so there are, intuitively, grounds for optimism about single global-grid algorithms even though we present no extensions of the theory to the non-self-adjoint problems we consider. The main disadvantage in condensing out intermediate scales is that the coarse grid, on which all optimal approaches require an exact solve, cannot necessarily become as coarse as one might like.

The field of locally uniform mesh refinement is spanned by a continuum of resolution strategies governed by clustering rules which control the size and shape of the refined subregions. Global refinement lies at one extreme and pointwise adaptive refinement at the other. As soon as the global tensor product mesh is abandoned a host of difficult practical decisions need to be made about data structures and clustering algorithms. The logic required to handle the numerous types of subgrid-subgrid interactions which can arise and to insure the consistency of the data structure is a significant impediment to efficient parallelism. In contrast, "horizontal" neighbor-neighbor interactions are simple. The sufficiency of a two-level approach in obtaining reasonable convergence is demonstrated in section 4. Compelling superiority of approaches with a greater richness of scales has not yet been fully established in production parallel software, although it may be ultimately,

Experience on parallel computers gained from a two-level approach will be beneficial in any event.

3. Iterative domain decomposition algorithms

As mentioned in the introduction, preconditioned iterative methods and domain decomposition provide a framework suitable for the description of a wide class of algorithms. The four common elements of this framework are: a global operator arising from the discretization of the PDE (or system of PDEs); an approximate inverse, or preconditioner, for the global operator; an iterative method relying only on repeated application of the preconditioned operator; and a geometry-based partition of the discrete unknowns so that size, locality, and uniformity can be exploited in applying the preconditioned operator. Since the numerical analysis literature contains many successful discretization schemes and iterative methods specialized for different operator properties, such as the presence or absence of definiteness and symmetry, the recent burgeoning effort in iterative domain decomposition algorithms has concentrated primarily (though not exclusively) on the interaction of the second and fourth of these elements. In the parallel context, this is a natural preoccupation because the bottleneck to parallelism usually (though not exclusively) lies in the requirement of the global transport of information in the preconditioner.

Many of the numerical examples described in section 4 rule out the use of iterative methods based on symmetry, but permit the assumptions of definiteness and diagonal-dominance. In particular, full or incomplete factorizations of subdomain matrices can be undertaken without pivoting. Because of its robustness, we join many recent users [13, 32, 38, 42] in adopting the parameter-free generalized minimum residual (GMRES) method [37] as the outer iteration. The main disadvantages of GMRES, its linear and quadratic (in iteration index) memory and execution time requirements, respectively, must be mitigated by scaling and preconditioning. For other acceleration schemes, such as Chebyshev, the memory and execution time requirements may be only constant and linear, respectively, but GMRES dispenses with the difficulty of estimating parameters. The primary type of decomposition used herein involves roughly unit aspect ratio tiles, as opposed to thin strips. Ordering the interior points (and the physical boundary points other than cross-points) first, the cross-points last, and the interfaces connecting the cross-points in between, gives a nested-dissection-like "arrow" matrix appearance to the global discrete operator, which we denote A. The basic structure of our preconditioner B is the block-upper triangular portion of the arrow matrix. The application of B^{-1} thus begins with a cross-point solve, which updates the right-hand sides of a set of independent interface solves. These, in turn, update the right-hand sides of a set of interior solves. For a nine-point stencil, the cross-point result would also update the interior right-hand sides. However, there is no dependence, within a single iteration, of the interface solution upon the result of the interior solution, or of the cross-point solution upon either. (In [11], structurally symmetric arrow matrix preconditioners were compared against the corresponding triangular forms on a variety of strip-wise decomposed problems. It is found therein that retaining the interior-to-interface coupling in the preconditioner generally reduces the total number of iterations required to attain a fixed convergence criterion, but that the execution time of the structurally symmetric algorithm is greater, because of the cost of the extra set of subdomain solves in each iteration. The first and second sets of subdomain solves are inherently sequential.)

The derivation of the coefficients of the preconditioner blocks is as follows. The cross-point equations are simply a scaled coarse grid discretization of the continuous PDE. Physical boundary points lying at tile corners are retained in the cross-point system in order to accommodate first-order Neumann or mixed conditions in this coarse grid discretization. Weighted averaging possibilities for the derivation of the coarse grid operator arise from the possession of the coefficients and right-hand side on finer grids surrounding each cross-point, but these are not currently exploited. The current implementation supports LU-based Gaussian elimination on the coarse-grid system. This solve is the chief parallel bottleneck in the preconditioner and can be performed in either of two ways:

redundantly on each processor after broadcasting the required coefficient data for small systems, or in a fully distributed fashion for large systems. Determination of the most efficient technique is generally domain and network dependent. If strip decompositions are used, there is no cross-point system, and the lower-right block of the preconditioner is simply the interface system described below.

The tile interior equations consist of fine grid discretizations of the PDE over local regions, with physically appropriate boundary conditions along any true boundary segments and Dirichlet boundary conditions at artificial interfaces. Only first-order differences are accommodated in the physical boundary conditions of the preconditioner, even if higher-order are employed in the operator A. The current implementation supports full LU Gaussian elimination, incomplete LU decomposition, or modified incomplete LU decomposition. Each tile performs its interior solve completely independently.

Unlike the coarse grid and tile interior equations, which bear the physical dimension of the underlying PDE and have natural preconditionings, the lower-dimensional interfacial equations are properly derived from a related pseudo-differential operator, a theoretically well-developed approach we do not pursue here because of the difficulty in applying it to arbitrary problems. Instead, we have compared three approaches referred to below as (a) tangential, (b) truncated, and (c) interface probe. The tangential interface preconditioner is the one-dimensional discretization of the terms of the underlying operator which remain when the derivatives normal to the interface are set to zero. The truncated interface preconditioner is a discretization of the full underlying operator, with the coefficients associated with non-interfacial unknowns set to zero. The interface probe preconditioner has been described elsewhere [9, 29] as a low-bandwidth approximation to the true capacitance matrix of the interfacial unknowns in the ambient matrix corresponding to the degrees of freedom of the interface itself and the two subdomain interiors on either side.

The differences between these three techniques are perhaps most easily visualized by considering the example of Laplace's equation on a uniformly discretized square partitioned by an interface parallel to one pair of edges into subdomains 1 and 2, the interfacial unknowns being subscripted 3. Let A_{11} and A_{22} be the subdomain operators, let A_{13} and A_{23} translate the values on the interface into the respective subdomain boundary condition right-hand side vectors, and vice versa for A_{31} and A_{32}. The tangential preconditioner is the tridiagonal matrix with diagonal elements -2 and sub- and super-diagonal elements 1. The truncated preconditioner is the same except for -4's on the diagonal. The interface probe preconditioner is the truncated preconditioner minus a diagonal matrix whose elements are those of the vector $[A_{31}A_{11}^{-1}A_{13} + A_{32}A_{22}^{-1}A_{23}]e$, where e is the vector of all 1's. The probe preconditioner has the same row sum as the actual Schur complement matrix for the interface, namely $A_{33} - A_{31}A_{11}^{-1}A_{13} - A_{32}A_{22}^{-1}A_{23}$. These three preconditioner matrix types differ only along the diagonal, with the elements of the probe diagonal lying between the first two.

4. Numerical experiments

The numerical experiments of this section serve to illustrate the effectiveness of the domain decomposition methods employed in terms of the convergence of the iterations and also the effectiveness of the locally uniform mesh refinement in terms of the convergence of the discretization.

4.1. Model problems

We present twelve model problems, each containing a single dependent variable and two independent variables. These restrictions on the number of variables beg generalization, so we comment briefly at the outset. Multiple dependent variable cases have been examined for stripwise decompositions in [29] and will be presented for the current cross-point of the algorithm in a subsequent paper oriented towards applications. The extension of our current techniques to three-dimensional problems is straightforward, but not necessarily effective. Optimal or near optimal algorithms for

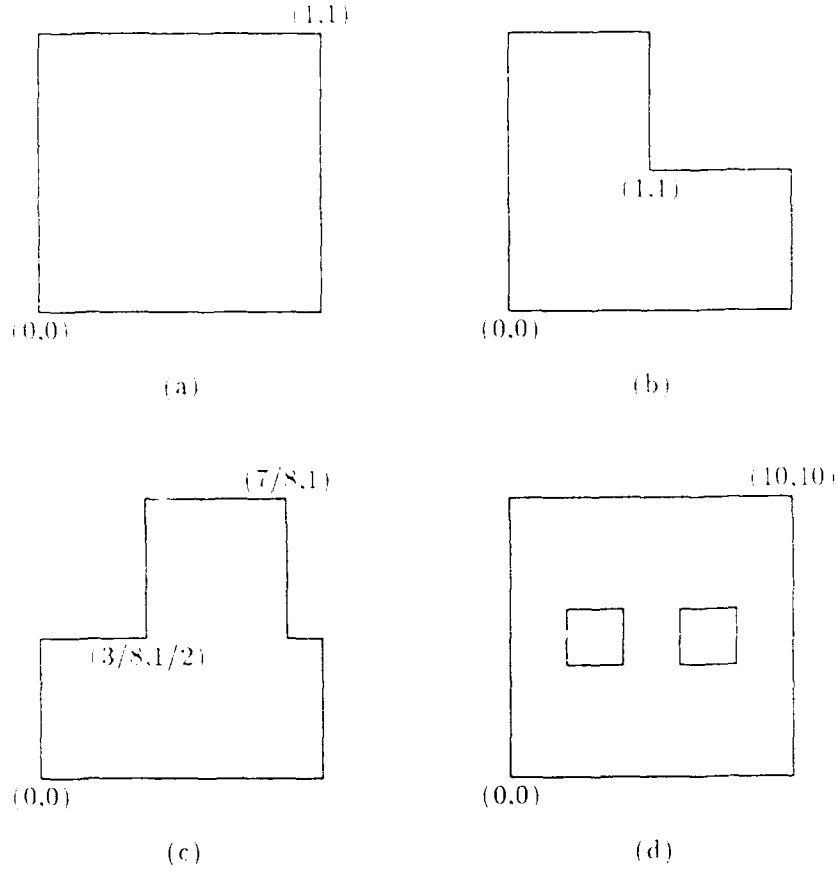


Figure 5: The four domains considered in this paper.

three-dimensional problems are known which require a more implicit lower-right corner block of the preconditioner, containing more than cross-points alone [15, 17]. We have not yet examined these so-called "wire-basket" forms of the preconditioner. From a parallel perspective, they introduce additional sequential overhead, and a careful consideration of the trade-offs between convergence rate and cost per iterations will be required in a future study.

Some of the problems below are self-adjoint and could be discretized in a symmetric manner and perhaps solved more cheaply with conjugate gradients than with GMRES. Our main interest, however, is in the more extensible formulation. In all the examples to follow except for the last, an exact solution of the continuous problem $\mathcal{L}u = f$ is specified. From this u, all of the following source terms f and boundary condition inhomogeneities g may be calculated. In cases where the expressions for f and g are sufficiently simple, they are written out along with the solution. The twelve problems include four different domains, pictured in Figure 5.

The first two examples, with constant coefficients and an exact solution quadratic in each independent variable, are extremely simple and possess truncation-error-free second-order finite difference representations. They are identical except for the type of boundary conditions along one side of their square domain. These problems are not candidates for mesh refinement; rather, they are chosen to show the deterioration in convergence rate caused when Dirichlet boundary conditions are replaced with Neumann, and to allow controlled experimentation on the effect of inaccurate boundary conditions in the preconditioner. The poor convergence of #2 using the preconditioner of #1 led to the decision to expand the cross-point system to include physical boundary points in the general case.

Problem #1: Pure isotropic diffusion with all Dirichlet boundaries.

$$\nabla^2 u = 4$$
$$u(x, y) = x^2 + y^2$$
Dirichlet data on $\partial\Omega$
$$\Omega = \text{Unit square}$$

Problem #2: Pure isotropic diffusion with a partial Neumann boundary.

$$\nabla^2 u = 4$$
$$u(x, y) = x^2 + y^2$$
Dirichlet data on the three lower sides of $\partial\Omega$
$$\frac{\partial u}{\partial n}(x, 1) = 2$$
$$\Omega = \text{Unit square}$$

The next example is included to study orientation-sensitivity of the substructuring due to anisotropic diffusion, for comparison with problem #1, to which it is identical when $a = 1$. The order-of-magnitude ratio between the diffusion coefficients in the x and y directions is mathematically indistinguishable at the discrete level from an order-of-magnitude physical domain aspect ratio in an isotropic problem.

Problem #3: Anisotropic diffusion.

$$\frac{\partial}{\partial x}\left(a\frac{\partial u}{\partial x}\right) + \frac{\partial^2 u}{\partial y^2} = 2(a + 1)$$
$$u(x, y) = x^2 + y^2$$
$$a = 10$$
Dirichlet data on $\partial\Omega$
$$\Omega = \text{Unit square}$$

The fourth example is a prototype convection-diffusion problem: a passive scalar in a plug flow which is fully developed at the outflow. It is a companion problem to #2 in the sense of possessing a smooth solution with one Neumann boundary, but asymmetry due to the convection. In that its anisotropy comes from a first-order operator, it is also an interesting complement to #3.

Problem #4: Plug-flow convection-diffusion with fully-developed outflow boundary.

$$-\nabla^2 u + c\frac{\partial u}{\partial y} = f$$
$$u(x, y) = \sin(\pi x)\sin\left(\frac{\pi y}{2}\right)$$
$$c = 10$$
$u = 0$ on the three lower sides of $\partial\Omega$
$$\frac{\partial u}{\partial n}(x, 1) = 0$$
$$\Omega = \text{Unit Square}$$

The next two examples (the first two from the standard "population" of elliptic problems in [35, 36]) bring in non-constant coefficients, the latter in a non-self-adjoint way with Robin boundary conditions.†

Problem #5: Self-adjoint, non-constant coefficient, Dirichlet boundaries.

$$\frac{\partial}{\partial x}\left(e^{xy}\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(e^{-xy}\frac{\partial u}{\partial y}\right)-\frac{u}{1+x+y}=f$$
$$u(x,y)=e^{xy}\sin(\pi x)\sin(\pi y)$$
$$u=0 \text{ on } \partial\Omega$$
$$\Omega=\text{Unit square}$$

Problem #6: Non-self-adjoint, non-constant coefficient, Robin boundaries.

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial}{\partial y}\left((1+y^2)\frac{\partial u}{\partial y}\right)-\frac{\partial u}{\partial x}-(1+2y+y^2)\frac{\partial u}{\partial y}=f$$
$$u(x,y)=0.135(e^{x+y}+(x^2-x)^2\log(1+y^2))$$
$$u-\frac{\partial u}{\partial n}=g \text{ on } \partial\Omega$$
$$\Omega=\text{Unit square}$$

The derivative is the outward normal.

The seventh example, from [1, 27], has a smooth solution, but rapidly varying coefficients along an internal layer. Here, the solution itself gives no hint of the requirement of mesh refinement. Interestingly, the locations of maximum error in a uniformly refined discretization of the PDE do not even occur at the internal layer itself, but towards the interiors of the two subdomains it divides [23].

Problem #7: Internal layer.

$$\nabla a\nabla u=f$$
$$a(x,y)=1+b\arctan\left(x-\frac{1}{2}\right)+c\arctan\left(d\left(y-\frac{1}{2}\right)\right)$$
$$u(x,y)=16x(1-x)y(1-y)$$
$$b=0.65,\ c=0.35,\ d=10.0$$
$$u=0 \text{ on } \partial\Omega$$
$$\Omega=\text{Unit square}$$

The next three examples are obtained by taking three different values of the convection, respectively $c=0$, $c=-10$, and $c=1$, in the convection-diffusion problem below.

Problems #8–10: Cylindrically-separable reentrant corner convection-diffusion problem

† The more widely available reference [25] contains an identical listing of problem #5 and a similar but not identical version of #6. A typographical error in the latter renders it ill-posed.

$$-\nabla^2 u + \frac{c}{r}\frac{\partial u}{\partial r} = 0$$

$$u(x,y) = r^{\alpha}\sin\left(\frac{2}{3}\left(\theta - \frac{1}{2}\pi\right)\right)$$

where $r = \sqrt{(x-1)^2 + (y-1)^2}$

and $\theta = \arg((x-1) + i(y-1))$, $0 \le \theta < 2\pi$

Dirichlet data on $\partial\Omega$

Ω = L-shaped region

The first of these corresponds to pure diffusion, and the second and third to convection in towards the reentrant corner, and away from it respectively, at a rate inversely proportional to radius. The respective values of the radial eigenfunction exponent α are $\frac{2}{3}$, $\frac{1}{3}$, and approximately 10.0442, from the Euler equation formula $\alpha = \left[c + \sqrt{c^2 + \frac{16}{9}}\right]/2$. The first two solutions of this trio lack derivatives at the reentrant corner. The last is everywhere twice-differentiable, but the solution is characterized by steep variation in the three *non*-reentrant corner regions, where $r > 1$. Local mesh refinement is critical to improving the accuracy of a finite-difference solution. In addition to refinement, a simple change to the finite difference scheme in the vicinity of the reentrant corner is made that substantially improves the accuracy of the solution; this is described in more detail in [21].

The eleventh example, from [4, 27], illustrates how an irregularly-shaped domain may force a minimum granularity upon a tessellation comprised of congruent tiles. For the problem at hand, the minimum granularity is near the ideal one.

Problem #11: T-shaped domain.

$$\nabla^2 u = 4 - 2\cos(y)e^x$$

$$u(x,y) = x^2 + y^2 - xe^x\cos(y)$$

Dirichlet data on $\partial\Omega$

Ω = T-shaped region

The last example, from [7], is provided to illustrate the accommodation of non-simply-connected domains. Again, the geometry imposes a minimum granularity on congruent tiles.

Problem #12: Two-hole domain.

$$-\frac{\partial}{\partial x}\left(\left(1 + \sin\frac{\pi x}{10}\right)\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\left(1 + \sin\frac{\pi x}{10}\sin\frac{\pi y}{10}\right)\frac{\partial u}{\partial y}\right) = x^2 + y^2$$

$u = 0$ on outer boundary of $\partial\Omega$

$$\frac{\partial u}{\partial y}(x,0) = 0 \text{ on hole boundary of } \partial\Omega$$

Ω = Two-hole region

Perspective surface plots of the solutions to these twelve problems are given in Figures 6 and 7.

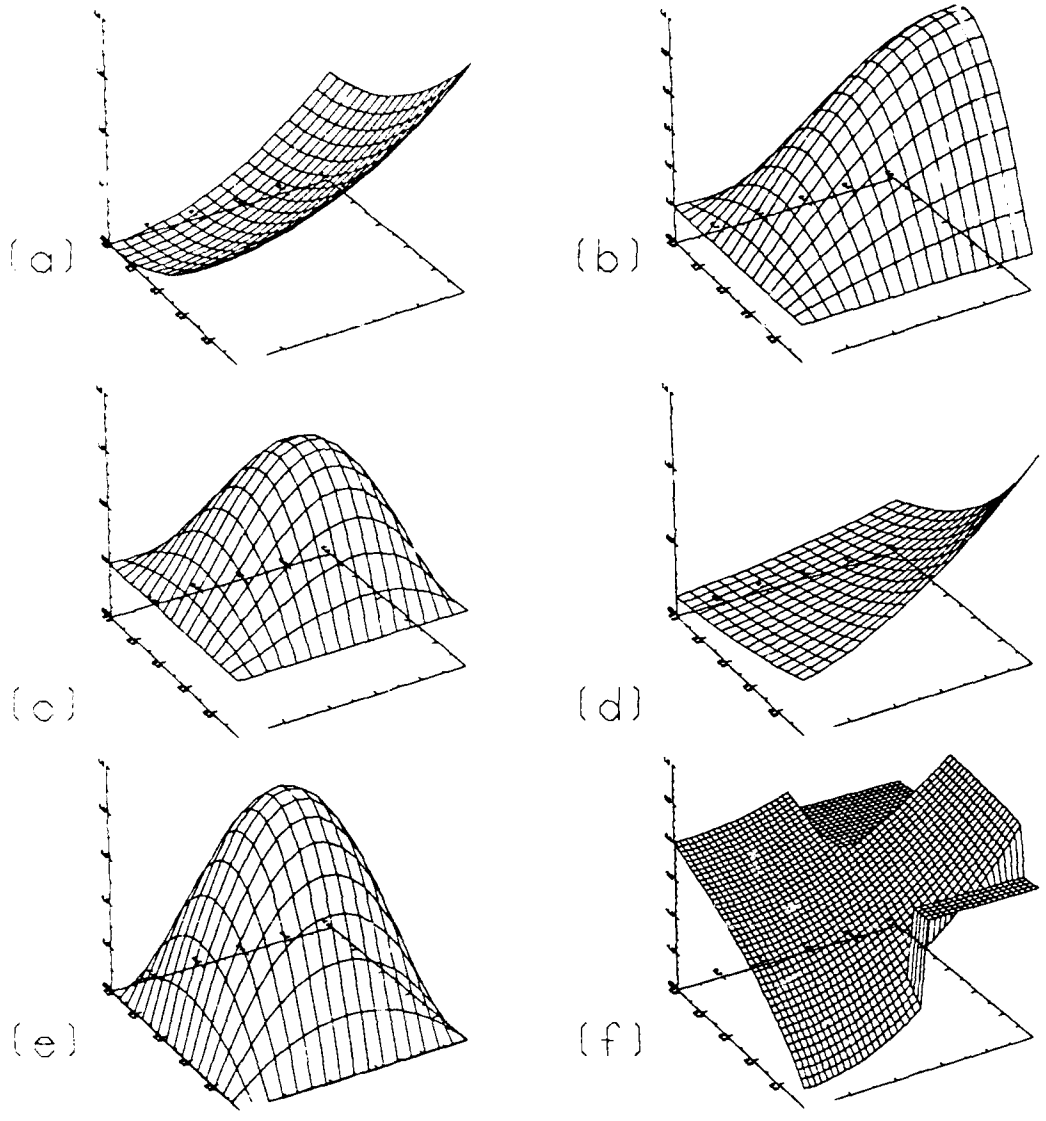


Figure 6: Surface plots of the test problem solutions: (a) #1–3, (b) #4, (c) #5, (d) #6, (e) #7, (f) #11. (Note that the solution to #11 is smooth; the apparent fronts are due to zeroing the surface over the undefined regions of the T-shaped domain.)

4.2. Parameters studied

Several categories of experiments are reported. First, a two-dimensional parameter space consisting of coarse grid resolution and overall (uniform) resolution is explored by numerical experiment on problems #1–10. A non-restarted GMRES algorithm is used, block-triangularly preconditioned with exact solves on the subdomain interiors and on the coarse grid, and with tangential interface solves. Here, as throughout this study, we use exclusively right preconditioning and an initial iterate of zero. The goal of these experiments is the evaluation of the algorithm over a range of resolutions, in terms of iteration count and execution time, for comparison with back-of-the-envelope complexity analyses.

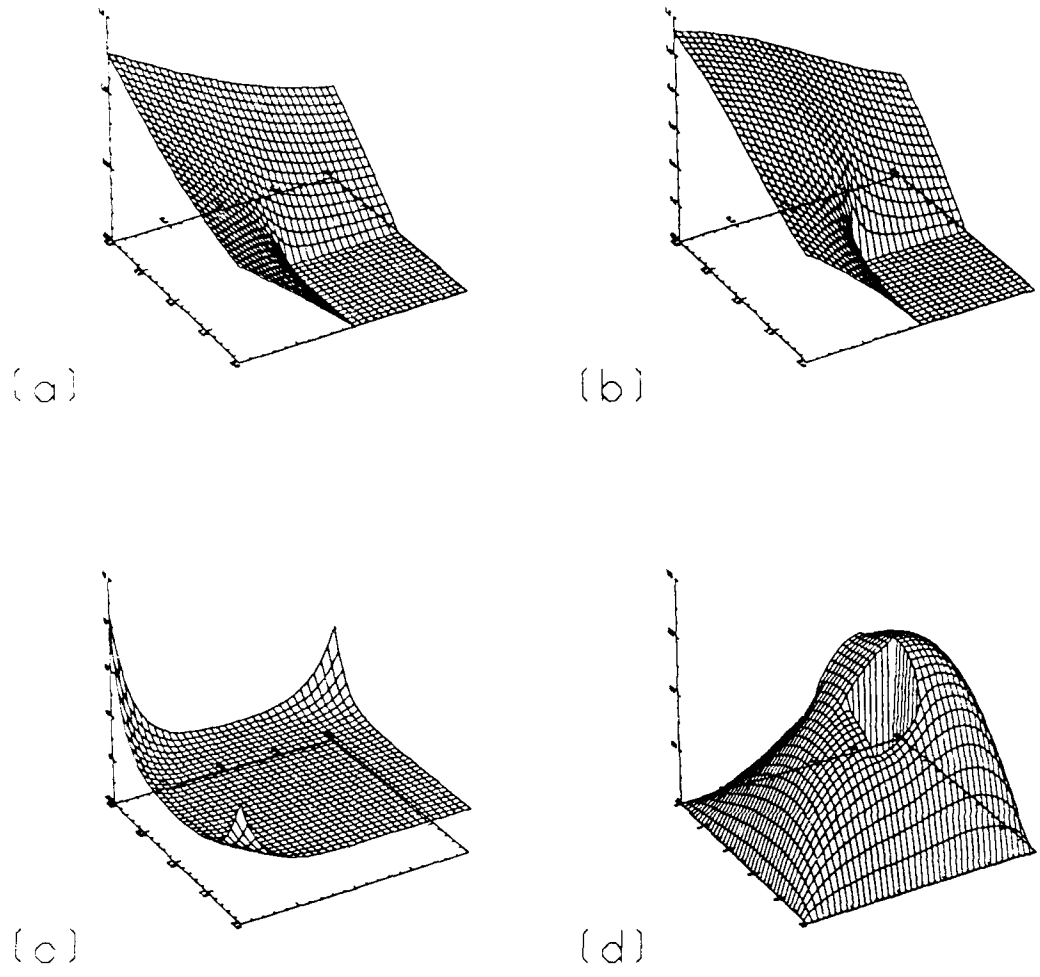


Figure 7: Surface plots of the test problem solutions: (a) #8, (b) #9, (c) #10, (d) #12.

Another set of experiments is performed on problems #7–10 with the goal of evaluating the economy of the local refinement technique. We show that local uniform mesh refinement is capable of significant CPU and memory savings with no sacrifice of accuracy relative to uniform refinement, but that improving the discretization in simple ways can be more effective than considerable refinement. In a third set, we evaluate the effect of decomposition orientation for non-unit-aspect ratio tiles, using problems #1–4. The limiting cases are the stripwise decompositions previously considered by us in [29]. In another, brief proof-of-concept section, we present results for the complex domain problems, #11 and #12. We then evaluate different preconditioner options than the exact interior solves and tangential interface solves used in all of the examples above. With exact interior solves, we compare three different interfacial preconditioners, and for tangential interface solves, we compare three different interior preconditioners. Finally, we compare our preferred options in this set to global incomplete factorizations for all of the problems which are posed on square domains.

t	m	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	128	1	4	1	2	1	5	1	NA	NA	NA
2	64	7	15	13	22	22	20	20	12	11	4
4	32	11	24	18	28	37	35	35	17	16	12
8	16	13	23	24	30	39	32	32	23	22	18
16	8	10	17	22	27	31	25	23	16	19	16
32	4	7	13	16	20	24	17	15	11	12	10
64	2	-	-	-	-	-	-	-	-	-	-
128	1	1	1	1	1	1	1	1	1	1	1

Table 1: Iteration count as a function of number of tiles per side of circumscribing square, t, and number of mesh points along a tile side, m, at constant refinement parameter, $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}. The last two lines of the table are not available experimentally due to minimum discrete subdomain size conventions in the code; however, the last line consists of all 1's by definition, when $t = h^{-1}$.

The timings given below are from a Multiflow Trace 14/200 computer using 64-bit reals. All of the code (primarily in C but with FORTRAN computational kernals) was compiled with the default (-O3) optimization and with version 2.1.3 of the compilers. Because of the varying performance of hardware (vector, parallel, superscalar) on different problem sizes (due to different startup costs and data dependency limitations), execution times are difficult to compare directly. The reader should keep in mind while studying the results that different organizations of the code and different compiler capabilities can account for large variations in times across architectures and software releases. We have run the same experiments (to the extent supported by memory) on two other Unix machines and find that the proportion of time spent in factorization and solution phases varies widely between machines even though the relative rankings of total timings remain mostly the same. In addition, replacing the nonsymmetric bandsolver in LINPACK used to solve the linear systems with a custom nonpivoting routine produces a large benefit on one computer (a factor of three reduction in time), but has little effect on another.

4.3. Convergence as a function of coarse grid granularity

In order to test coarse grid granularity over a large range, we fix the finest mesh spacing at $h^{-1} = 128$ (relative to the length of the domain, whether that be 1 in the first seven problems, or 2 in the next three) and investigate the tradeoff between numbers of tiles and points per tile, as shown in Tables 1 and 2 and plotted in Figure 8. The mesh is identical and uniform for all runs in these tables (with the obvious exception that one quadrant of it is not present in the L-shaped domain problems, #8–10, which therefore lack single-tile entries). The convergence criterion is a relative reduction in residual of five orders of magnitude. Table 1 shows that the iteration count peaks in the middle of the granularity range, at either 4 or 8 tiles per side. The bottom row of all 1's can be supplied without benefit of actual experiments, since it represents a direct solve on a single grid. The top row entries differ from 1 in problems where the preconditioner has different (lower order) boundary conditions than the operator A.

Table 2 shows the deceptiveness of iteration count alone as a measure of overall performance. In execution time, the extreme runs, representing single-domain limiting cases, suffer due to the high cost per iteration, even though the number of iterations required is very small. This table is a profound illustration of the title of [10]: *Domain Decomposition Beneficial Even Sequentially*. The most favorable total *sequential* execution times are found for multi-domain cases near the iteration

t	m	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
1	128	416	419	416	418	424	432	425	NA	NA	NA
2	64	108	115	113	121	121	119	117	84	85	81
4	32	30	38	34	41	48	46	45	25	25	23
8	16	12	19	20	25	34	27	25	14	14	12
16	8	7	13	23	23	42	27	23	9	14	12
32	4	17	32	41	55	70	47	35	36	21	17

Table 2: Execution time (sec) as a function of number of tiles per unit length, t, and number of mesh points along a tile side, m, at constant $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}.

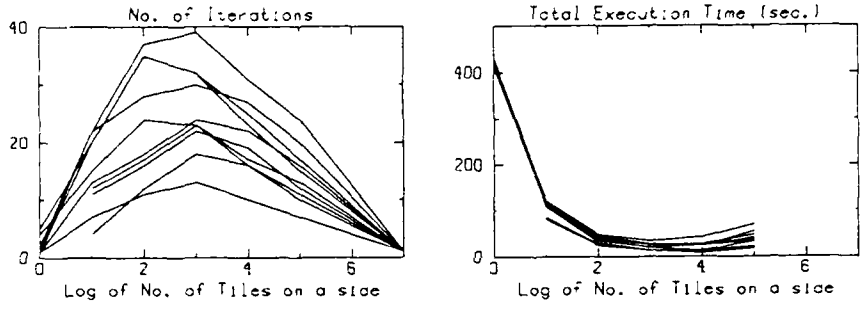


Figure 8: Plots of Tables 1 and 2 (problems #1–10 superposed), illustrating that the minimum execution time serial algorithm occurs near $t = 16$ tiles on a side, despite the large iteration count at this granularity.

count maxima, in particular at 16 tiles per side.

The factorization of the banded matrix in the single subdomain case is the dominant contribution to the overall time. In problems #1–7, over 410 seconds are spent doing the factorization alone. Of course, one might not ordinarily employ exact solves on the single domain cases, although many structural analysis codes do this very thing. A comparable penalty will accrue in an attempt to do exact solves on a very fine "coarse" grid, in which each tile contains just one point. However, the table of execution times is truncated beyond tile sizes of $m = 4$.

The behavior in Table 2 can be understood with reference to back-of-the-envelope complexity estimates for the solution and factorization operators of the preconditioner. We observe that there are $O(t^2)$ cross-point, interfaces, and interiors. Naturally ordered banded direct factorizations and solves require $O(Nb^2)$ and $O(Nb)$ operators respectively, where N is the number of unknowns and b the bandwidth. For the cross-point system, $N \approx t^2$ and $b \approx t$; for the interfaces, $N = m$ and $b = 1$; and for the subdomain interiors, $N = m^2$ and $b = m$. Thus, the interface operation counts are always asymptotically subdominant and can be omitted in the following. From choosing the larger of the cross-point and interior complexities, we see that factorization costs $\max_{(t,m)}\{O(t^4), O(t^2m^4)\}$ and solves cost $\max_{(t,m)}\{O(t^3), O(t^2m^3)\}$. Since $m = 128/t$ in these experiments, the first term grows with t and the second decays with it. Quick calculations reveal that (to the resolution

t	h^{-1}	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
2	16	6	10	10	12	12	15	10	6	5	3
4	32	11	19	16	17	24	31	21	12	11	11
8	64	12	20	22	25	29	28	23	17	14	16
16	128	10	17	22	27	31	25	23	16	13	16

Table 3: Iteration count as a function of number of tiles per side of circumscribing square, t, and refinement parameter, h^{-1}, at constant number of mesh points along a tile side, $m = 8$, for a reduction in the initial residual of 10^{-5}.

of the table) the minima for both factorization and solve costs occur at or between $t = 16$ and 32 when $h^{-1} = 128$. The tendency of buffer overhead, neglected in the estimates, is to favor a slightly smaller number of tiles t than thus estimated. It is important to note that the memory requirements follow the solve complexities above. Thus, for a fixed memory size, an intermediate coarse grid granularity accommodates the largest problem in core. Of course, all of these per iteration complexity estimates need to be redone when the preconditioner blocks are other than exact solves, for instance, incomplete factorizations. However, incomplete and exact factorizations differ little in actual cost per iteration when the grid is narrow enough in the rapidly ordered direction, which includes the case of small, square tiles.

4.4. Convergence as a function of tile refinement

In contrast to the previous section, we here investigate iteration count as a function of overall resolution, for a fixed number of subintervals per tile. The results are shown in Table 3. The global mesh grows in refinement from 16 to 128 as the number of points per tile remains constant at 8. In spite of the fact that the truncation error improves with at least h^{-1}, we use the same convergence tolerance of 10^{-5} as in the earlier tables. The fine grid in the last line of Table 3 corresponds to the $t = 16$ case of the earlier tables.

The experiments suggest that the iteration count is bounded nearly independently of h, and thus that the two-level algorithm is nearly optimal asymptotically in the constant m limit. In fact, some of the finest mesh results are even relatively *better* than preceding coarser ones. This should not be regarded as surprising, since there is a steep price for this favorable iteration count when m is held constant and h^{-1} is increased, namely, a larger cross-point system. We have not pursued any theoretical justification for this bound, but the theory for conjugate gradient iteration for self-adjoint problems, see, *e.g.*, [6, 40], contains similar results, namely, constant upper bounds on the iteration count for constant m.

As representative convergence histories, we present Figure 9 which follows the residual reduction over five orders of magnitude, and the time versus iteration count history for problems #1 and #2. The latter plots reveal the quadratic term in the GMRES work estimate that comes from the need to orthogonalize each iterate over a subspace whose size grows linearly in iteration count. This pair of figures also illustrates the poorer conditioning of Neumann problems, since the initial iterates and the solutions converged to are identical, and so are the operators except for one Neumann boundary segment.

4.5. Economies of local mesh refinement

Examples #7 through #10 allow us to display the well-known benefits of local uniform mesh refinement in elliptic problems: comparable accuracy in considerably fewer operations, compared with global uniform refinement. We solve these problems at refinement levels of $h^{-1} = 32, 64, 128$, and 256, based on the global grid, but perform both global and local refinements for comparison, where possible. (The finest global refinement does not fit into the memory available, which is, of

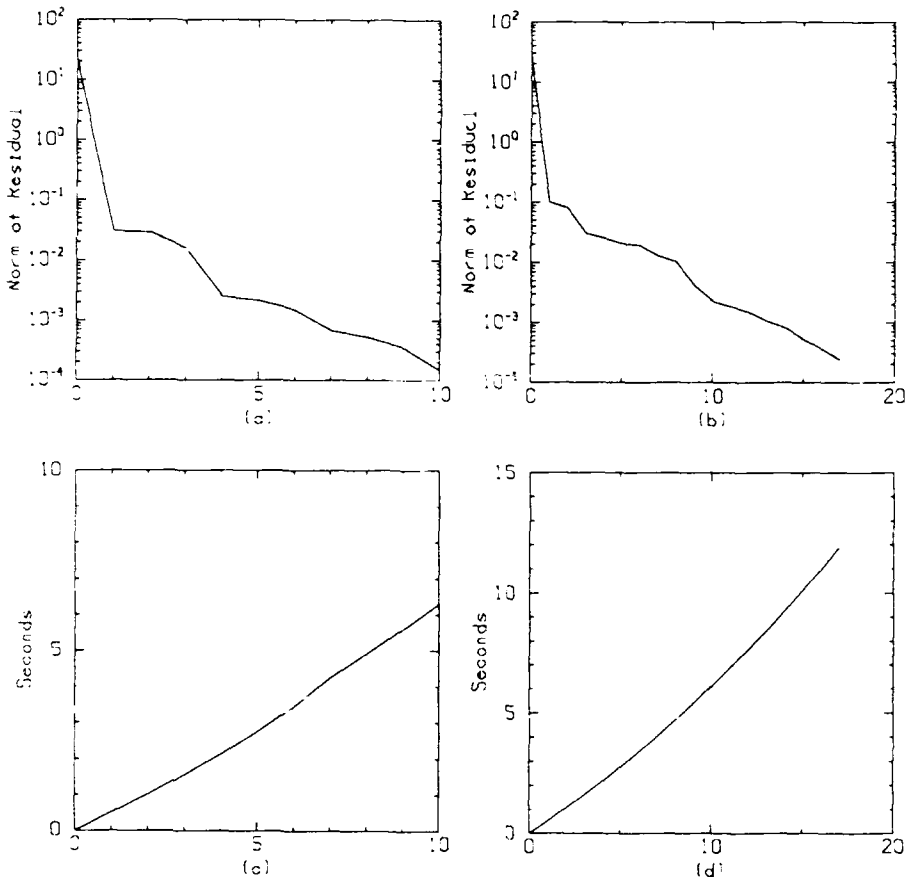


Figure 9: Convergence histories for problems #1 and #2, for $t = 16$, $m = 8$, $h^{-1} = 128$. (a) and (b) show the normalized Euclidean norm of the residual versus iteration count, and (c) and (d) show time versus iteration count.

course, another of the main motivations for LUMR, along with execution time savings.) All of these computations were made with a reduction in the residual of 10^{-8}, so that the measure of the error would not be contaminated by the residual. In all cases, the choice of where to refine is made by hand. In a forthcoming paper [23] we will show that the local error is not always adequate as an indicator of the optimal refinement location. Since we are interested in studing how domain decomposition and mesh refinement interact, given a good refinement strategy, we eliminate the latter question from this study.

Tables 4 through 7 compare global refinement results on the left, and local on the right. Each set of columns lists the number of unknowns, the sup-norm of the error, the number of iterations to reduce the discrete residual by 8 orders of magnitude, and the total execution time thus required. The right-most column gives the execution time ratios for each refinement level. Memory use ratios can also be estimated from the tile structure of the discrete problem, but the present code records no explicit allocation measurements. All entries share a constant value of $t = 8$ in order to fix regions of enhanced refinement that do not shrink as h does. Therefore, the "global" iteration columns of Tables 4 through 7 comprise a convergence study which is complementary to both Table 1 (in which h is constant) and Table 3 (in which m is constant).

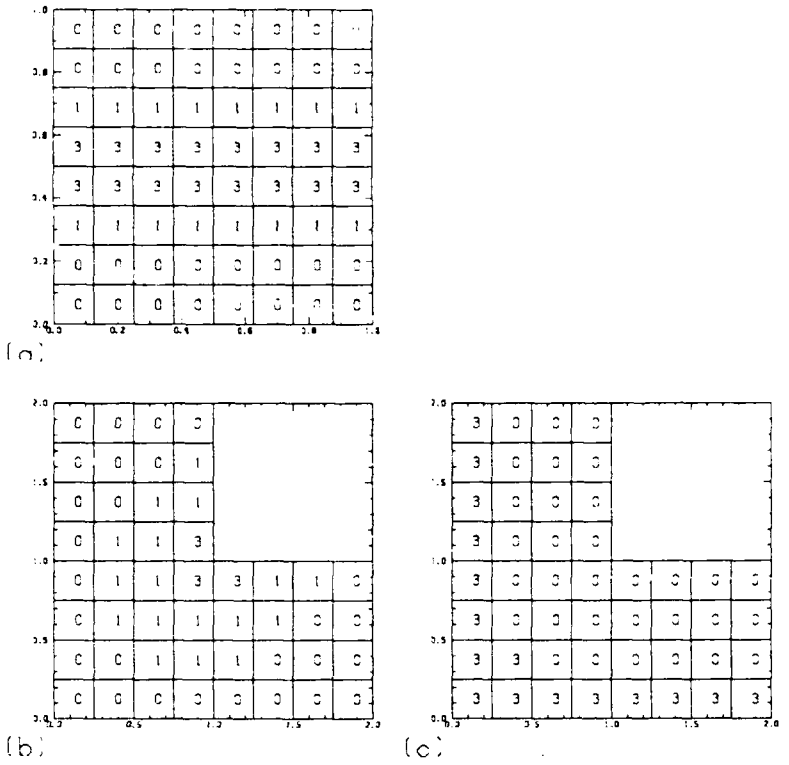


Figure 10: Refinement levels. The maximum (third level) local uniform refinements. (a) Problem #7. (b) Problems #8 and #9. (c) Problem #10. In second level tests, all tiles showing "3" are set to "2". In first level tests, these are further reduced to "1". In zeroth level refinement, all tiles are set to "0", which here corresponds to $m = 8$.

		Global				Local				Ratio
h^{-1}	m	N_G	ϵ_G	I_G	T_G	N_L	ϵ_L	I_L	T_L	T_G/T_L
32	4	1089	1.58(-4)	26	3.9	1089	1.58(-4)	26	3.9	1.00
64	8	4225	3.95(-5)	37	10.9	2641	4.15(-5)	46	12.2	.89
128	16	16641	9.89(-6)	53	51.1	5729	2.06(-5)	65	31.5	1.62
256	32		NA	NA	NA	18049	1.70(-5)	80	99.1	NA

Table 4: Number of unknowns N, sup-norm of the error ϵ, iteration count I, and execution time T (sec) for problem #7 (internal layer), globally and locally refined, along with execution time ratios, for a reduction in the initial residual of 10^{-8}.

The behavior of iteration count with each doubling of global refinement in the self-adjoint problems in Tables 4 and 5 is consistent with the logarithmic growth in conditioning with h^{-1} proved for self-adjoint problems in [6]. The locally refined examples also worsen in conditioning with h^{-1} when t is held constant, but the CPU time advantage of local refinement increases with h^{-1}, overall.

		Global				Local				Ratio
h^{-1}	m	N_G	ϵ_G	I_G	T_G	N_L	ϵ_L	I_L	T_L	T_G/T_L
32	4	834	1.30(-2)	24	2.7	834	1.30(-2)	24	2.7	1.00
64	8	3202	8.30(-3)	32	6.8	1818	8.30(-3)	35	6.2	1.10
128	16	12546	5.25(-3)	41	27.1	2410	5.26(-3)	37	7.6	3.57
256	32		NA	NA	NA	4746	3.33(-3)	41	16.4	NA

Table 5: Number of unknowns N, sup-norm of the error ϵ, iteration count I, and execution time T (sec) for problem #8 (reentrant corner, pure diffusion), globally and locally refined, along with execution time ratios, for a reduction in the initial residual of 10^{-8}.

		Global				Local				Ratio
h^{-1}	m	N_G	ϵ_G	I_G	T_G	N_L	ϵ_L	I_L	T_L	T_G/T_L
32	4	834	6.97(-2)	23	2.6	834	6.97(-2)	23	2.6	1.00
64	8	3202	5.65(-2)	37	8.2	1818	5.66(-2)	34	5.7	1.44
128	16	12546	4.53(-2)	40	26.1	2410	4.58(-2)	37	7.6	3.43
256	32		NA	NA	NA	4746	3.67(-2)	41	16.5	NA

Table 6: Number of unknowns N, sup-norm of the error ϵ, iteration count I, and execution time T (sec) for problem #9 (reentrant corner, convective inflow), globally and locally refined, along with execution time ratios, for a reduction in the initial residual of 10^{-8}.

		Global				Local				Ratio
h^{-1}	m	N_G	ϵ_G	I_G	T_G	N_L	ϵ_L	I_L	T_L	T_G/T_L
32	4	834	7.35(-1)	22	2.4	834	7.35(-1)	22	2.4	1.00
64	8	3202	4.15(-1)	28	5.7	1610	4.30(-1)	25	3.6	1.58
128	16	12546	2.19(-1)	34	21.5	4698	2.40(-1)	29	8.5	2.53
256	32		NA	NA	NA	17018	1.98(-1)	35	51.6	NA

Table 7: Number of unknowns N, sup-norm of the error ϵ, iteration count I, and execution time T (sec) for problem #10 (reentrant corner, convective outflow), globally and locally refined, along with execution time ratios, for a reduction in the initial residual of 10^{-8}. The error values here appear large, but are in fact small relative to the size of the solution.

The sup-norm of the error shows sublinear improvement in h in problems #8 and #9, as one expects with non-differentiable solutions. The second-order accuracy of the discretization is readily apparent (the ratio of errors is almost exactly 4 with each reduction of h by 2) in problem #7, and the first-order accurate treatment of convection in problem #10 leaves its signature as well.

In Table 8 we show the benefit of rediscretization of the tiles surrounding the reentrant corner in problems #8 and #9 to fit the discrete solution to the known power-law radial dependence of the singular exact solution (see the problem statements above). Rather than making the customary Taylor series assumptions, we take $u(r) = u_0 + ar^p + br^{2p}$, where p is derivable from a local analysis (*see* [21]). Figure 11 displays $u(r)$ along the ray $\theta = \frac{5\pi}{4}$, which is the symmetry axis of the three

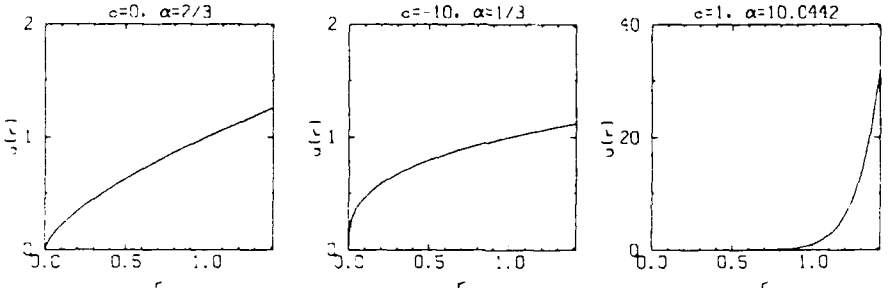


Figure 11: Cross-section of $u(r)$ along the symmetry axis. (a) Problem #8, pure diffusion, non-differentiable at $r = 0$. (b) Problem #9, convective inflow, strengthening the singularity. (c) Problem #10, convective outflow, eliminating the singularity.

		Problem #8				Problem #9			
h^{-1}	m	ϵ_A	I_A	T_A	ϵ_A/ϵ_L	ϵ_A	I_A	T_A	ϵ_A/ϵ_L
32	4	1.63(-3)	23	2.5	.13	2.16(-2)	21	2.2	.31
64	8	1.04(-3)	61	13.5	.13	1.88(-2)	35	5.8	.33
128	16	6.66(-4)	64	16.5	.13	1.61(-2)	36	7.0	.35
256	32	4.26(-4)	67	28.5	.13	1.33(-2)	39	15.1	.36

Table 8: Sup-norm of the error ϵ, iteration count I, and execution time T (sec) for problems #8 and #9 locally refined with asymptotic fitting, along with the ratio of the error to the corresponding local entries without asymptotic fitting in Tables 5 and 6.

L-shaped problems.

4.6. Numerical compromises associated with domain geometry

The domains of problems #11 and #12 provide an interesting test of the tile decompositions advocated herein because they can be more simply described with less restrictive decompositions. For instance, if the only restriction on the decomposition was that all subdomains had to be rectangular, the first has a two-subdomain, and the second a five-subdomain decomposition. In contrast, our uniform-size decompositions require a minimum of 48 and 23 tiles respectively. However, because the Neumann boundary conditions of #12 require a minimum stencil width for the coarse grid solve in the preconditioner, we must further bisect (in each coordinate direction) obtaining a 92-tile decomposition. Convergence results for some constant h discretizations are given in Tables 9 and 10.

Though domain geometry prohibits much exploration of granularity parameter space, we note that: (a) the practical granularities are in the range found most useful for problems #1–10 in Tables 1 and 2; (b) the number of processors available in a typical medium-scale parallel computer (say 2^5 through 2^8) is appropriate for tessellating shapes such as these, which, when allowed to

t	m	I	T
8	16	14	8.8
16	8	12	6.9
32	4	10	16.8

Table 9: Execution time (sec) as a function of number of tiles per unit length, t, and number of mesh points along a tile side, m, at constant $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5} on problem #11 (T-shaped domain). There are 12546 unknowns.

t	m	I	T
10	16	99	176
20	8	82	239
40	4	70	1124

Table 10: Execution time (sec) as a function of number of tiles on a side, t, and number of mesh points along a tile side, m, at constant $h^{-1} = 160$ subintervals on a side, for a reduction in the initial residual of 10^{-5} on problem #12. (Note that the two-hole domain of this example is in $[0, 10] \times [0, 10]$). There are 24001 unknowns.

Sub	Int	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Exact	Tang.	13	23	24	30	39	32	32	23	22	18
Exact	IP(0)	78	90	92	85	93	293	86	55	55	47
Exact	Trun.	92	165	171	96	157	—	136	66	66	56
ILU(0)	Tang.	—	—	—	—	—	—	—	—	—	243
ILU(1)	Tang.	—	—	—	—	—	—	—	290	287	153
MILU(0)	Tang.	38	—	61	—	63	—	72	42	42	41

Table 11: Iteration count for different preconditioner block combinations at constant refinement parameter, $h^{-1} = 128$, and tessellation, $t = 8$, $m = 16$, for a reduction in the initial residual of 10^{-5}. GMRES was restarted after every 100 iterations. — indicates that the iteration had not converged after 500 steps.

undergo quasi-uniform distortion, are sufficiently general for a large class of typical two-dimensional engineering applications; and (*c*) the quasi-uniform tiles represent quasi-uniform quanta of work for a convenience in load-balancing that the less restrictive minimum tessellations do not have.

It should be noted that these problems are alternatively solved very successfully by embedding into the circumscribing squares, and using preconditioners based on fast solves on the squares, in what is known as the capacitance matrix method (see, *e.g.*, [34]). Complex domains are often better candidates for embedding preconditioners than for decomposition preconditioners in terms of the size of the capacitance system (see, *e.g.*, [8]). We note that either approach can lead to effective parallelism, since readily parallelized fast solvers exist [12].

4.7. Tests of algorithmic combinations

Tables 11 and 12 explore different algorithmic combinations for the preconditioner blocks.

Sub	Int	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
Exact	Tang.	13	17	18	22	30	24	24	13	12	10
Exact	IP(0)	77	96	99	87	101	317	89	35	35	26
Exact	Trun.	98	166	174	105	156		135	45	45	30
ILU(0)	Tang.								-		178
ILU(1)	Tang.				-			--	224	220	197
MILU(0)	Tang.	23		47		49		62	20	20	19

Table 12: Execution time (sec) for different preconditioner block combinations at constant refinement parameter, $h^{-1} = 128$, and tessellation, $t = 8$, $m = 16$, for a reduction in the initial residual of 10^{-5}. GMRES was restarted after every 100 iterations. -- indicates that the iteration had not converged after 500 steps.

Four subdomain preconditioners (exact, ILU(0), ILU(1), and MILU(0)) and three interface preconditioners (IP(0), truncated, and tangential), as described in section 3, are tested on a standard tessellation for problems #1–10. Many combinations did not converge after 100 iterations so a *restarted* GMRES [37] was employed, in which an intermediate solution was computed, and a new Krylov subspace begun every 100 iterations. Up to five restart cycles were attempted. (Note that problems #1–7 contain 16641 unknowns, and #8–10 contain 12546, so a union of subspaces consisting of a maximum of 500 search directions represents only 3 to 4 percent of the dimensionality of the problem; even so, it is beyond the range of attractive performance for such methods.)

Evidently, the interface probe technique IP(0) does not work as well as the tangential preconditioner on these problems, though it is always better than the truncation preconditioner. A possible explanation for the poor performance of the IP(0) interface handling is that probing near the cross-points is an inaccurate characterization of the mutual influence of points on intersecting interfaces. Though IP(0) is a good technique for adapting interface preconditioning to coefficient variation, the tables illustrate that the straightforward version for stripwise decompositions is ineffective on a cross-point problem with "short" interior interfaces. Suitable generalizations of the interface probe technique are important to applications because the information required to construct IP(0) is embedded directly into the matrix elements of the linear system to be solved, whereas construction of the tangential preconditioner requires information about the original differential operator, and the relevant collection of terms is not defined for source-sink operators.

We note that the non-exact subdomain solves perform very poorly for these problems, relative to the exact solves. The exception is MILU(0), which performs well on the Dirichlet problems. Figure 12 is a useful diagnostic for the poor performance of many of the combinations. Shown in the six panels is a surface plot of the elements of the vector $B^{-1}f$ for problem #1, decomposed into an 8×8 array of 16×16 tiles. A good preconditioner B will yield a plot resembling the actual solution of the problem, $u = x^3 + y^2$ (see Figure 6(a)). This is reasonably well approximated by combinations of tangential interface preconditioning and exact or MILU subdomain preconditioning. The other combinations with tangential interface preconditioning show that the "wire-basket" part of the solution is well-defined, but that the subdomain preconditioning is poor. ILU(1) is slightly superior to ILU(0), as expected, and as more bands of fill-in are permitted, ILU(k) eventually converges to an exact solve (when $k = m - 1$ for the five-point operator). The combinations with IP(0) and truncated interface preconditionings show that the quality of the preconditioning is lost at the "wire basket" stage, independent of the subdomain solves.

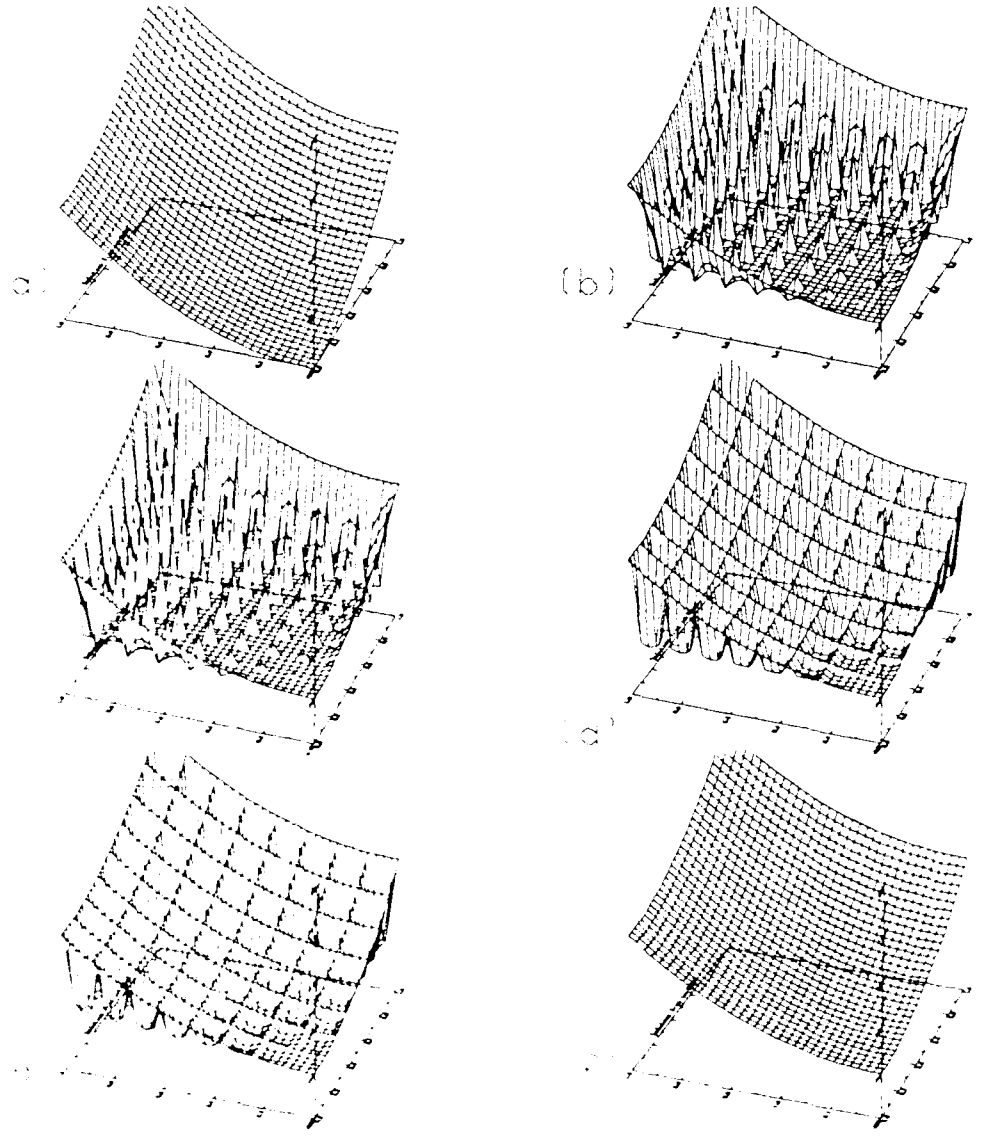


Figure 12: Surface plots of $R^{-1}f$ for the six combinations of *Table 12 on Problem #2*: (a) Exact Tangential, (b) Exact ILP0, (c) Exact Truncated, (d) ILU(0) Tangential, (e) ILU(1) Tangential, (f) MILU(0) Tangential.

4.8. Convergence dependence on decomposition orientation

Tables 13 and 14 provide a link between the present experiments and the stripwise decomposition of Table 1 therein. Tangential and ILP(0) preconditioners are computed on problem #2 for stripwise and unit aspect rectangular point decompositions, as parameterized by the number of blocks in the x and y directions, l_x and l_y.

Three main results stand out from these tables. First, we again note the weakness of ILP(0) relative to the tangential preconditioner in most cases, and most dramatically in the many processor scaling case of 8×8 and 16×16 tiles. Secondly, we note that, for the same number of processors (e.g., comparing the (2,2), (1,4) and (4,1) triad, or the (4,4), (1,16), and (16,1) triads), the stripwise decompositions yield smaller runtimes even though they usually require more iterations. This

		Tangential				IP(0)			
t_x	t_y	#1	#2	#3	#4	#1	#2	#3	#4
1	1	1	4	1	2	1	4	1	2
1	2	6	10	4	3	10	10	9	17
2	1	6	12	10	7	10	11	10	16
2	2	7	15	13	22	14	17	17	22
1	4	8	14	6	5	12	13	9	18
4	1	8	17	16	8	12	12	18	15
4	4	11	24	18	28	30	35	38	38
1	8	12	20	8	9	16	20	11	21
8	1	12	21	28	8	16	19	30	16
8	8	13	23	24	30	78	90	92	85
1	16	20	34	12	17	25	34	17	32
16	1	20	43	52	8	25	31	51	21
16	16	10	17	22	27	156			

Table 13: Iteration count as a function of number of tiles per horizontal and vertical sides of circumscribing square, at constant refinement parameter, $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}. GMRES was restarted after every 100 iterations.

is due to the narrower bandwidths of the strip subdomain bandsolvers, and is an effect that *would be less pronounced* had we used fast FFT-implementable, sparse, or incomplete factorization solvers. Finally, we note that stripwise decompositions can exploit anisotropies. In problem #3, for instance, keeping the subdomains unbroken along the strongly coupled direction in the PDE (the x direction) leads to better iteration counts and execution times than otherwise. However, cross-point decompositions are a good compromise between the "good" and the "bad" strip orientations. In problems in which the optimal strip orientation either varies from location to location or is unknown *a priori*, a cross-point decomposition employing the same number of subdomains is a good alternative to strips.

4.9. Comparison with undecomposed alternatives

Tables 15 and 16 provide a more realistic comparison between global and decomposition approaches to preconditioning than Tables 1 and 2. Recall that requiring exact bandsolves on subdomains penalizes beyond reason the performance of the lesser decomposed alternatives. Here we use two standard preconditioners, MILU(0) and ILU(1), in place of exact solves on the same version of problems #1–7.

All cases involving an unscaled Neumann or Robin boundary condition (the exact Neumann problems) fail to converge in 500 steps using the global MILU approach, and one of these also confounds the ILU preconditioning. The tile-based preconditioning, lubricated by diagonal scaling, converged in a relatively modest number of iterations for all problems. The MILU preconditioning leads to the best execution times in the cases for which it works (namely, the non-Dirichlet problems). More importantly for future needs, the tile-based approach has *significantly better prospects for parallel execution*, and *since it is competitive with the global approach* in overall parallel efficiency, relative to the *best serial algorithm of any listed* will be relatively so. This will be true even on large, distributed memory machines with relatively slow interprocessor communication, since the amount of communication required in domain decomposition methods is

		Tangential				IP(0)			
t_x	t_y	#1	#2	#3	#4	#1	#2	#3	#4
1	1	416	419	416	418	416	419	416	418
1	2	107	109	105	105	109	109	109	111
2	1	107	111	109	108	109	110	109	113
2	2	108	115	113	121	111	113	112	116
4	4	28	31	28	27	30	30	29	33
4	1	28	33	32	28	30	30	33	31
4	4	30	38	34	41	40	44	47	47
1	8	10	14	9	9	12	14	10	15
8	1	10	16	20	9	12	14	20	12
8	8	12	19	20	25	76	94	98	86
1	16	11	23	7	9	15	23	9	20
16	1	11	32	44	5	15	20	41	12
16	16	7	13	23	23	773			

Table 14: Execution time (sec) as a function of number of tiles per horizontal and vertical sides of circumscribing square, at constant refinement parameter, $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}. GMRES was restarted after every 100 iterations. indicates that the iteration had not converged after 500 steps.

Method	#1	#2	#3	#4	#5	#6	#7
Global/MILU	22		19		39		39
Global/ILU	45	71	61	67	59		56
Tile/Exact	13	23	24	30	39	32	32

Table 15: Iteration count for problems #1–7 for a global MILU(0)-preconditioned GMRES, a global ILU(1)-preconditioned GMRES, and an tile exact/tangential preconditioned GMRES (for $t = 8$, $m = 16$), at refinement parameter $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}.

Method	#1	#2	#3	#4	#5	#6	#7
Global/MILU	8		6		16		16
Global/ILU	19	37	30	34	28		26
Tile/Exact	12	19	20	25	34	27	25

Table 16: Execution time (sec) for problems #1–7 for a global MILU(0)-preconditioned GMRES, a global ILU(1)-preconditioned GMRES, and an tile exact/tangential preconditioned GMRES (for $t = 8$, $m = 16$), at refinement parameter $h^{-1} = 128$, for a reduction in the initial residual of 10^{-5}.

small, particularly compared to global techniques.

5. Conclusions and future directions

Experiments on a diverse group of problems demonstrate that a two-level domain decomposition algorithm with a single global coarse grid can provide "nearly" optimal convergence and allow

a great deal of flexibility in refinement strategy, while also permitting a data structure amenable to parallel and vector implementations, as summarized in closing below. Although often motivated by parallelization, domain decomposition may also yield runtime and memory use benefits as a sequential programming paradigm. Furthermore, the simple structure of individual blocks of the domain-decomposed preconditioner means that new applications are found for the "standard solvers" in conventional software libraries.

The traditional economies of local uniform mesh refinement can be straightforwardly incorporated into the domain decomposition framework at the price of interface handlers with conditionals for refinement differences between adjacent subdomains. Because of the highly modular nature of a standardized tile-oriented domain decomposition code, custom discretizations for certain classes of singularities may be archived into applications libraries for reuse.

The tile algorithm demonstrated herein in a superscalar mode on a Multiflow computer is amenable to vectorization in either of two ways. The regular operation sequences on the tensor-product subgrid arrays are precisely the type for which vectorizing compilers were conceived. The vector lengths depend on the precise form of solvers used in the preconditioner, but would tend to be rather small for the rows of individual 8×8 or 16×16 tiles found best in the two-dimensional applications above. An alternative form of vectorization can be realized by grouping together all tiles of a given (discrete) size and shape and operating in lock step on corresponding elements in each tile, assuming an identical solver is applied to each. A vector in this approach consists of the j^{th} element from each of the subdomains. Our 8×8 arrays of tiles would be thus be optimal for machines with a vector length of 64.

Parallelization requires careful attention to the load balancer/mapper and also to the coarse grid solve in the preconditioner. Some complexity estimates pertaining to alternative forms of the latter may be found in [24]. The main disadvantage of the two-level algorithm in the parallel context is that the choice of coarse grid granularity is even more of an "over-determined" problem than in serial. Communication cost per iteration and convergence properties potentially inveigh against the lower bounds imposed by domain geometry, solution and coefficient smoothness, and parallel load balance. The key determination for future applications of the tile methodology will be whether this over-determination is consistent in practice. Inasmuch as the examples herein are representative of single-independent variable problems, and parallel communication costs generally comprise a relatively *smaller* proportion of the total work in coupled multi-component problems, there are substantial grounds for optimism that this will be the case.

References

[1] O. Axelsson & B. Polman, *Block Preconditioning and Domain Decomposition Methods, II.* J. Comp. Appl. Math., 24(1988), pp. 55–72.

[2] I. Babuška, J. Chandra, and J. Flaherty eds., *Adaptive Computational Methods For Partial Differential Equations*, SIAM, Philadelphia, 1983.

[3] M. J. Berger & J. Oliger, *Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations*, J. Comp. Phys., 53(1984), pp. 484–512.

[4] P. E. Bjorstad & O. B. Widlund, *Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures*, SIAM J. Num. Anal., 23(1986), pp. 1097–1120.

[5] J. H. Bramble, R. E. Ewing, J. E. Pasciak & A. H. Schatz, *A Preconditioning Technique for the Efficient Solution of Problems with Local Grid Refinement*, Comp. Meths. Appl. Mech. Eng., 67(1988), pp. 149–159.

[6] J. H. Bramble, J. E. Pasciak & A. H. Schatz, *The Construction of Preconditioners for Elliptic Problems by Substructuring, I.* Math. Comp., 47(1986), pp. 103–134.

[7] ———, *Preconditioners for Interface Problems on Mesh Domains*, (manuscript).

[8] B. L. Buzbee, F. W. Dorr, J. A. George & G. H. Golub, *The Direct Solution of the Discrete Poisson Equation on Irregular Regions*, SIAM J. Num. Anal., 8(1971), pp. 722–736.

[9] T. F. Chan, Boundary Probe Domain Decomposition Preconditioners for Fourth Order Problems, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods*, SIAM, Philadelphia, 1989, pp. 168–172.

[10] T. F. Chan & D. Goovaerts, *Domain Decomposition Beneficial Even Sequentially*, Technical Report 88-18, UCLA Comp. and App. Math., June 1988.

[11] T. F. Chan & D. E. Keyes, *Spectral versus Probe Interface Preconditioning for Convective-Diffusive Problems*, Technical Report, Res. Inst. for Advanced Comp. Sci. (RIACS), NASA-Ames, 1989. (In preparation).

[12] T. F. Chan & D. Resasco, *A Domain-Decomposed Fast Poisson Solver on a Rectangle*, SIAM J. Sci. Stat. Comp., 8(1987), pp. s14-s26. (Errata: same journal, 8:457).

[13] A. Dervieux, L. Fezoui, H. Steve, J. Periaux & B. Stoufflet, Low-Storage Implicit Upwind-FEM Schemes for the Euler Equations, *Proceedings of the 11th International Conference on Numerical Methods in Fluid Dynamics*, Springer, Berlin, 1989, pp. 215–219.

[14] D. Dewey & A. T. Patera, Geometry-Defining Processors for Partial Differential Equations, B. J. Alder ed., *Architectures and Performance of Specialized Computer Systems*, Academic Press, New York, 1988.

[15] M. Dryja, A Method of Domain Decomposition for Three-Dimensional Finite Element Elliptic Problems, R. Glowinski, G. H. Golub, G. A. Meurant & J. Periaux ed., *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1988, pp. 43–61.

[16] ———, Optimal Iterative Refinement Methods, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods*, SIAM, Philadelphia, 1989, pp. 168–172.

[17] M. Dryja & O. B. Widlund, *Some Domain Decomposition Algorithms for Elliptic Problems*, Technical Report 438, Courant Institute, NYU, April 1989.

[18] ———, *On the Optimality of an Additive Refinement Method*, 1989. (Prepared for the Fourth Copper Mountain Conference on Multigrid Methods).

[19] R. E. Ewing & R. D. Lazarov, Adaptive Local Grid Refinement, *Proceedings SPE Rocky Mountain Regional Meeting*, 1988. SPE No. 17806.

[20] W. D. Gropp, *Local Uniform Mesh Refinement for Elliptic Partial Differential Equations*, 1983. (Submitted to SIAM J. Sci. Stat. Comp.).

[21] ———, *A Simple Finite Difference Approximation for the Laplacian near Reentrant Corners*, 1989. (In preparation).

[22] W. D. Gropp & D. E. Keyes, Domain Decomposition on Parallel Computers, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods*, SIAM, Philadelphia, 1989, pp. 260–268.

[23] ———, *Pointwise Error Estimates are Inappropriate for Mesh Refinement*, 1989. (In preparation).

[24] ———, *The Parallel Complexity of Tile-based Substructuring of PDEs with Local Mesh Refinement*, 1989. (In preparation).

[25] E. N Houstis, R. E. Lynch & J. R. Rice, *Evaluation of Numerical Methods for Elliptic Partial Differential Equations*, J. Comp. Phys., 27 (1978), pp. 323–350.

[26] H. Jarausch, *On An Adaptive Grid Refining Technique for Finite Element Approximations*, SIAM Journal on Scientific and Statistical Computing, 7/4 (1986), pp. 1105–1120.

[27] D. E. Keyes & W. D. Gropp, *A Comparison of Domain Decomposition Techniques for Elliptic Partial Differential Equations and their Parallel Implementation*, SIAM J. Sci. Stat. Comp., 8 (1987), pp. s166-s202.

[28] ———, Domain Decomposition Techniques for Nonsymmetric Systems of Elliptic Boundary Value Problems: Examples from CFD, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods*, SIAM, Philadelphia, 1989, pp. 321–339.

[29] ———, *Domain Decomposition Techniques for the Parallel Solution of Nonsymmetric Systems of Elliptic BVPs*, 1989. Appl. Num. Meths. (To appear).

[30] Y. Maday, C. Mavriplis & A. T. Patera, Nonconforming Mortar Element Methods: Application to Spectral Discretizations, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods*, SIAM, Philadelphia, 1989, pp. 392–418.

[31] S. McCormick & J. Thomas, *The Fast Adaptive Composite Grid (FAC) Method for Elliptic Equations*, Math. Comp., 45 (1986), pp. 439–456.

[32] A. Navarra, *An Application of GMRES to Indefinite Linear Problems in Meteorology*, Comp. Phys. Comm., 53 (1989), pp. 321–327.

[33] W. Proskurowski, Remarks on the Spectral Equivalence of Certain Discrete Operators, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods*, SIAM, Philadelphia, 1989, pp. 103–113.

[34] W. Proskurowski & O. B. Widlund, *A Finite Element-Capacitance Matrix Method for the Neumann Problem for Laplace's Equation*, SIAM J. Sci. Stat. Comp., 1 (1980), pp. 410–425.

[35] J. R. Rice, E. N. Houstis & W. R. Dyksen, *A Population of Linear Second Order, Elliptic Partial Differential Equations on Rectangular Domains - Part I*, Technical Report 2078, Mathematics Research Center, Univ. of Wisconsin - Madison, May 1980.

[36] ———, *A Population of Linear Second Order, Elliptic Partial Differential Equations on Rectangular Domains - Part II*, Technical Report 2079, Mathematics Research Center, Univ. of Wisconsin - Madison, May 1980.

[37] Y. Saad & M. H. Schultz, *GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM J. Sci. Stat. Comp., 7(1986), pp. 856-869.

[38] F. Shakib, T. J. R. Hughes, Z. Johan, Element-by-Element Algorithms for Nonsymmetric Matrix Problems Arising in Fluids, J. H. Kane ed., *Symposium on the Solution of Super Large Problems in Computational Mechanics*, Plenum, New York, 1989.

[39] B. Swartz, *Courant-Like Conditions Limit Reasonable Mesh Refinement to Order h^2*, SIAM J. Sci. Stat. Comp., 8(1987), pp. 924-933.

[40] O. B. Widlund, Iterative Substructuring Methods: Algorithms and Theory for Elliptic Problems in the Plane, R. Glowinski, G. H. Golub, G. A. Meurant & J. Periaux ed., *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1988, pp. 113-128.

[41] ———, Optimal Iterative Refinement Methods, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods*, SIAM, Philadelphia, 1989, pp. 114-125.

[42] L. B. Wigton, N. J. Yu & D. P. Young, GMRES Acceleration of Computational Fluid Dynamics Codes, *Proceedings of the AIAA 7th Computational Fluid Dynamics Conference*, 85-1494, AIAA, Washington, DC, 1985.

[43] H. Yserentant, *On the Multi-level Splitting of Finite Element Spaces for Indefinite Elliptic Boundary Value Problems*, SIAM J. Num. Anal., 23(1986), pp. 581-595.